Integrating the commercial software package LabVIEW with Fermilab's Accelerator Control NETwork.

W. BLOKLAND

Fermi National Accelerator Laboratory*

ABSTRACT

An overview is given of the developed interface between the in-house networking protocol, ACNET, and the commercially available software package for instrumentation control and analysis, LabVIEW. The interface supports data reading and writing access by consoles over ethernet and tokenring networks. In addition, LabVIEW applications can read and write data of other nodes by communicating over ethernet with the TCPORT server on a VAX. Tools are provided to help the developer implement LabVIEW programs and make entries in the ACNET database. A summary of applications and experiences is given.

INTRODUCTION

The Instrumentation Group of the Accelerator Division at Fermilab deals with the development and maintenance of sensors, actuators, and signal processing hardware, as well as making the measurements available for accelerator monitoring or control. Many of our tasks involve the use of computers. The tasks using computers can be subdivided in the following categories:

1) *Modeling:* Models are used to determine the behavior of an accelerator related process or to predict hardware performance for measuring accelerator parameters. Examples of these types of applications are beam position module frequency response, pulse propagation through hardware components, and filter calculations.
2) *Development:* During the development of a hardware component (e.g., a VXI card or an analog processor) a prototype setup is needed to verify the operation of the component.
3) *Maintenance:* Testing and calibration of equipment or other hardware can be a repetitive task; automation of these tasks saves time and money. An example of such a task is the calibration of an RF module, in which a signal generator supplies a test signal and a digitizing scope measures the quality of the RF module's processing.
4) *Accelerator Operation:* A monitoring or control system takes data and provides results to the operators or a supervisory program. Such a system is always on-line and must run reliably and communicate over the accelerator network.
5) *Accelerator Diagnostics:* To diagnose the operation or confirm a model of the accelerator, a data-acquisition system must be set up to take the required data. After the measurements are taken, the system is no longer needed and its components can be used for other purposes.

In looking for a solution that can handle all or most of these tasks efficiently, we derived the following requirements for a general purpose instrumentation platform:

1) the programming environment should support fast prototyping for one-time only uses, include extensive graphic display capabilities to make interpretation of data easy, and include an analysis library to support modeling;
2) control capability for a wide variety of instruments (e.g., VXI/VME, GPIB, CAMAC);
3) support for off-the-shelf software and hardware;
4) high reliability for on-line applications;
5) capability to communicate with control system.

While a VME embedded computer with the capability to communicate with the control system was available, this was not chosen as a general purpose platform for several reasons. To set up the embedded computer, a programming expert familiar with the details of the communication interface was required (the expert might or might not be available) and the system was difficult to use for fast prototyping because it lacked drivers for various instrumentation and had no graphics support or analysis library for data presentation and modeling. In addition, the cost of a VME/VXI crate had to be included even if the instrumentation to be automated was GPIB or CAMAC. Instead, the commercial package LabVIEW was chosen because it does support fast prototyping and graphics display, comes with hardware interface to most instruments, including software drivers, and includes an extensive analysis library. At the time, LabVIEW was only available on the Macintosh computer, which was also the desktop platform in the Division, resulting in the

* Operated by the Universities Research Association under contract with the U.S. Department of Energy.

Macintosh/LabVIEW combination as the instrumentation platform. The advantages of selecting a desktop computer as the platform for running instrumentation applications is both the availability of software and hardware and that only one computer is needed to perform the regular personal desktop duties as well as bench-top instrumentation applications. The main disadvantage is that embedded computers typically have real-time operating systems designed for fast and reliable data-acquisition and control applications. Choosing a desktop like the Macintosh will mean that certain applications cannot be implemented.

To add the capability of communicating with the control system, an interface to the ACcelerator NETwork (ACNET) was written. The integration of LabVIEW with ACNET is described in the following section, followed by an overview of applications and experiences.

INTEGRATION OF LABVIEW WITH ACNET

Communication

The ACNET protocol associates data with physical devices, for example, the value of the voltage of a power supply. In LabVIEW, an ACNET device is associated with an analysis result, a variable, rather than a device. To retrieve data, the ACNET protocol requires that remote nodes have a task called RETDAT to handle requests for data readings, and a task called SETDAT to handle data settings. On the LabVIEW nodes, the tasks are implemented in C and run as completion routines. The implemented RETDAT task supports single-reply, multi-reply interval-based, and multi-reply event-based on a single message containing requests for data from one or more devices. The SETDAT task supports a single-set of data on a single message containing requests for data settings of one or more devices. The RETDAT and SETDAT tasks talk to a server task that was already written for the Macintosh as part of the Lineac control system (see [1]). The server connects the tasks to a tokenring or ethernet network and provides ACNAUX (ACNET AUXiliary) node statistics services, such as which tasks are connected, the number of messages handled, and uptime. The data accessible by RETDAT and SETDAT is stored in shared memory that the LabVIEW program accesses through a library of read/write routines for the different variable types (see [2]).

From the operator's point of view, the LabVIEW/ACNET interface makes a LabVIEW node look like just another node in the accelerator network, so that the standard console environment can be used to read out or control a LabVIEW application.

As more complex applications were built, the ability to be read or be set was extended with a new interface that enables a LabVIEW program to read and set devices on other nodes, for example, the reading out of beam energy or current store number, or the setting of a timing gate. This interface, TCPORT, connects LabVIEW as a client to the VAX TCPORT server which will, on behalf of the client, issue the proper ACNET requests. The data requested from nodes must be scaled according to the database entries. To avoid having to reimplement these data scaling services and database lookup facilities already available on the VAX, the TCPORT server performs these functions, including permission checks and activity logging, before requesting the settings or sending scaled data to the TCPORT client on the Mac. Because the client is completely implemented in LabVIEW itself, it can also run on the other computers on which LabVIEW runs.

Because the instrumentation platform uses a desktop computer, additional commercial communication software is readily available. Utilities like Timbuktu (PC/MAC) and PlanetX (X-Windows) can remotely control a node before an ACNET console application has been written. These utilities display the screen of the remote Macintosh and give complete control as if it were the local computer. The utilities are also used after the system has become operational for diagnostic purposes by the system expert because it gives much more control over the node than a console application. Other often used communications utilities are Appleshare and FTP programs to up/download data and program files.

Database

The Fermilab control system utilizes a centralized database. When a control console makes a request for data for a particular device, the database matches the name of the device to look up on which node (e.g., the Mac running LabVIEW), the device resides and the SubSystem Device Number (SSDN) required by the node to identify the device. In addition, the database retrieves information about the size of the data and on how to transform and scale the data once retrieved from the node. To correctly retrieve the data, the database and the node must have matching information about the device. This is implemented by using one source, a spreadsheet table, to generate entries for the database and initialize the node. The table can be created and edited by a commercially available spreadsheet program like Excel. Each row represents a device entry, with each column representing an attribute of the device (see table 1).

6/16/94	Demo	Version 1.2										W. Blokland
Device name	SSDN #	Read/ Write	Var type	Bytes/ elem	Elem/ device	Initial value	Update rate	Com Unit	Eng Unit	Long/ Short	Nota tion	Description
T:DEMRAT	1	RW	I32	4	1	50	60	unit	step	short	dec	Demo step ratio
T:DEMSTP	2	RW	I16	2	1	1	60	unit	step	short	dec	Demo step number
T:DEMVAL	3	RW	SGL	4	1	1	60	unit	step	short	dec	Demo value
T:DEM	4	R	SGL	4	4000	0	120	unit	step	short	dec	Demo array
T:DEMSTR	6	R	STR	1	256	Hello	120	unit	step	short	dec	String
T:DEMTMP	7	R	SGL	4	1	85	600	Fahr	step	short	dec	TGC202 Room Temperature

Table 1. The device table used to initialize the LabVIEW/ACNET interface.

The table is used to initialize the ACNET interface each time the LabVIEW program is started. By doing this from a table instead of hardcoding this into the program, modifications to the device table do not require a recompilation of the code or involvement of an ACNET programming expert. A LabVIEW utility, Ac_MakeDabbel, is used to convert the device table to a database file with device entries. This way, a developer does not have to be familiar with the syntax of the database entries. The format of the device table is also suitable as documentation for what devices are defined for the node.

Development tools

To help a developer use the ACNET interface, a template of a LabVIEW/ACNET application is available. The template includes the initialization and termination of the ACNET interface and has a location on the diagram where the developer can insert the LabVIEW application (see figure 2). The developer only has to fill out the name of the device table, over which network the interface should communicate (or even both), and the name of the project. The template includes an error logging display and a statistics page. The statistics page shows the registered devices, the ACNET access activity as well as computing load on the node and the program versions of the ACNET interface. To help install the required software for the ACNET interface, an installer disk is available.

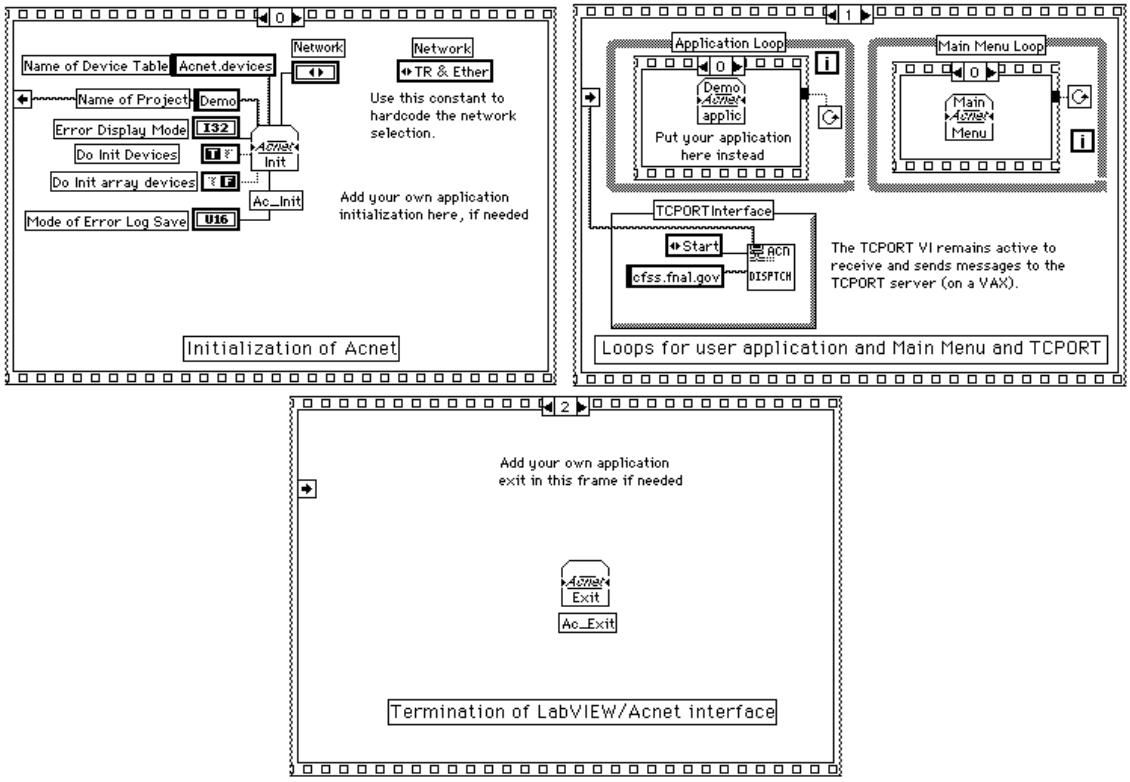

Figure 1. The Diagram of the Template for using the LabVIEW/Acnet interface.

To assist the development and documentation of LabVIEW system, the utility VI Hierarchy has been created (see figure 2). It displays the hierarchy of calling routines and the information about the operation of the routine. The utility allows the developer to browse through the hierarchy and print out documentation. Even though the graphical nature of LabVIEW automatically provides a graphof the data flow of each routine, a high level description of the functionality, much like the description of a c routine,is still needed to help others understand the program (see figure 3).

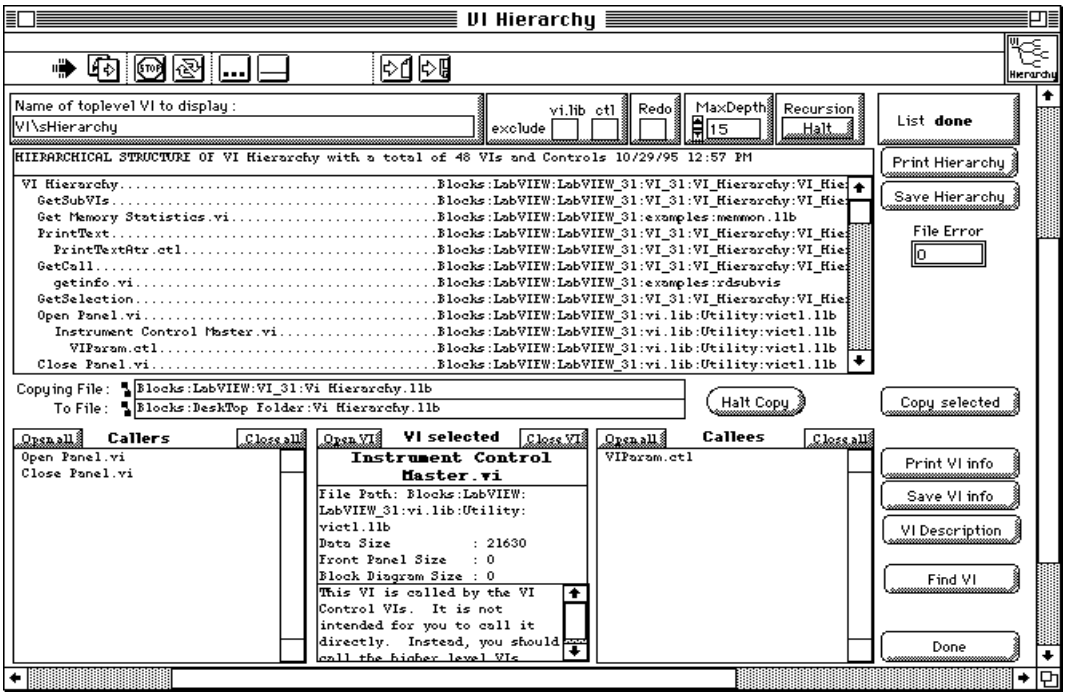

Figure 2. The Hierarchy Browser.

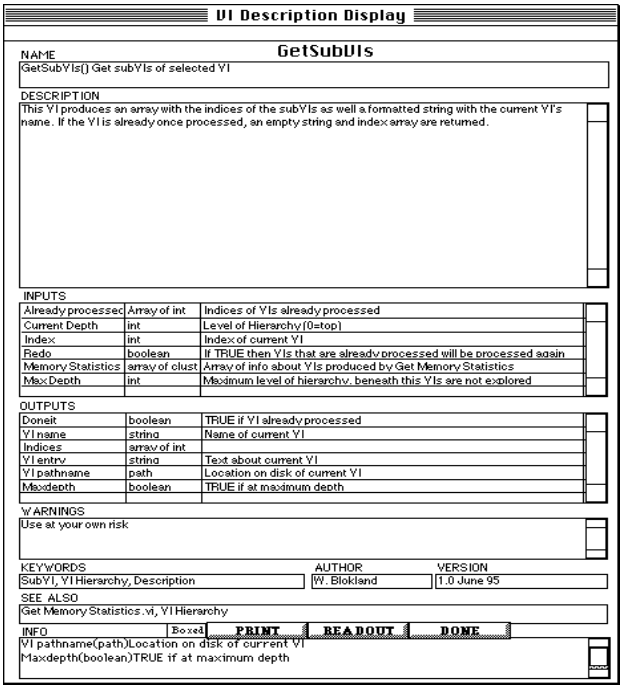

Figure 3. Documentation for a LabVIEW routine.

To make it possible to test or diagnose the LabVIEW application without requiring the use of the console environment, a LabVIEW utility, ParameterPage, was created to read out indexed devices, plot array devices, and time plot a device. The ParameterPage utility uses the TCPORT interface to request the data from any ACNET node (see figure 3). The TCPORT utility also makes it possible to prototype a console application in the LabVIEW environment before writing it in the console environment.

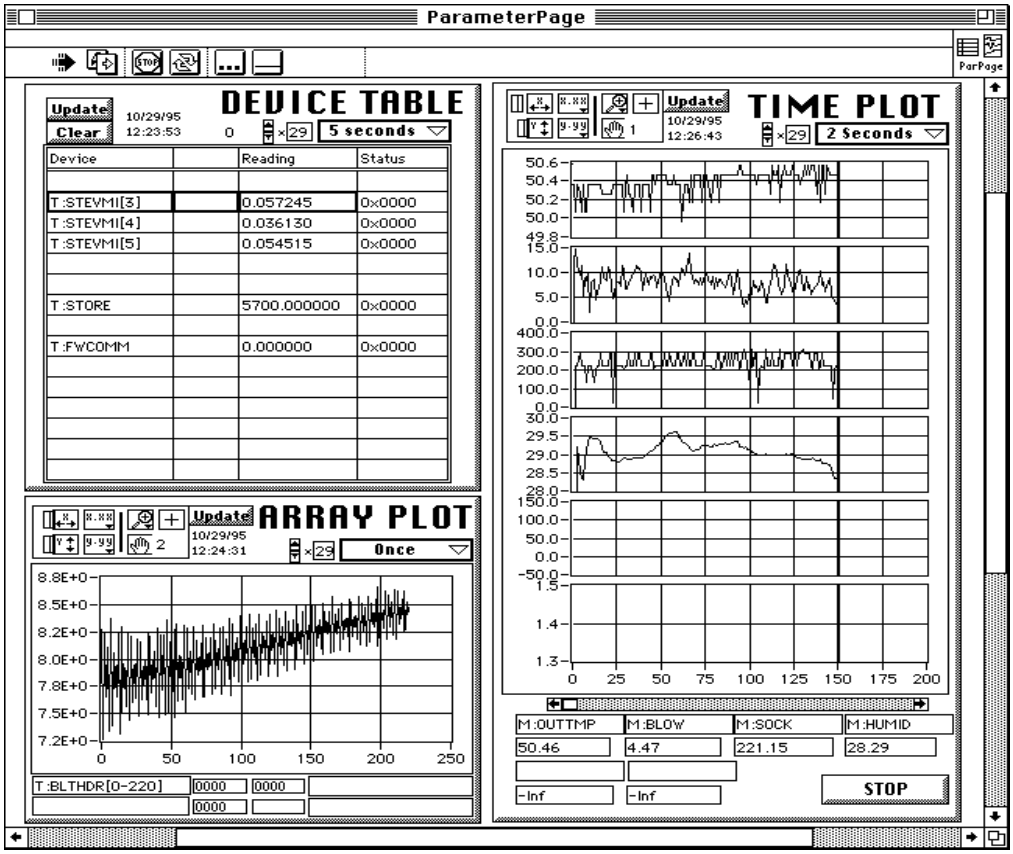

Figure 3. The LabVIEW parameter page.

In case a Macintosh system crashes and remote control is no longer possible over ACNET or with other means, the power to the computer can be cycled by an independent control channel using a CAMAC card. This proved to be very helpful during development and commissioning phases.

APPLICATIONS

A wide variety of applications have been made using LabVIEW. An overview of them is given in table 2. The table lists the name of and the use of the application. The next column of the table, labeled type, contains the abbreviations of instrumentation tasks listed in the introduction chapter. The operation of the application can be triggered by an accelerator event (e.g. beam injection), by the user, by a supervisory program, or the program repeating as fast as possible (cycle). The following two columns list the amount of data (in bytes) that must be acquired and how it is analyzed. The next two columns show the type of instrumentation and whether it was bought or developed in-house. The last column shows the time it takes to acquire, analyze, and present the data. This time gives a general idea about the time-frame of the applications, the times are not directly comparable because different computers are used (68000 versus PowerPC Macintosh). Specific applications are discussed in [3,4,5,6], and [7].

Application	Use	Type	Trigger	Data Amount	Analysis	Instrument	Com-mercial	response time
Beamline Tuner	Injection tuning of steering magnets	Ac Op	Injection Tevatron	5 kb	N-L damped oscill. fit	VXI digitizer	yes*	5 seconds
Synchrotron Light Monitor	Transverse emit-tance measurement	Ac Op	Cycle	100 kb	2-D non-lin gaussian fit	framegrabber/ HV supply	yes*	<1 s/bunch
Collision Point Monitor D0	D0 Collision point determination	Ac Op	Cycle	10 kb	Rectifying integrator	GPIB scope	yes*	90 sec^
Collision Point Monitor B0	B0 Collision point determination	Ac Op	Cycle	10 kb	Rectifying integrator	GPIB scope	yes*	75 sec^
Booster Ion Profile Monitor	Turn by turn transverse emittance	Ac Op	Injection Booster	2.5 Mb	Non-linear gaussian fit	VME digitizers	yes*	40 s/20k turns
MR Ion Profile Monitor	Turn by turn transverse emittance	Ac Op	Injection MR	8 Mb	Non-linear gaussian fit	VME digitizers	yes*	20 s/65k turns
Sampled Bunch Display	Long. emittance & int in MR/Tev	Ac Op	Cycle	10 kb	Integration & moments	GPIB scope	yes*	20 s/12 bunches
Accumulator Flying Wires	emittance measurement	Ac Op	User	20 kb	Non-linear gaussian fit	VME digitizers	yes*	20 sec/2 wires
Tevatron Flying Wires#	emittance measurement	Ac Op	Supervisory program	100 kb	Non-linear gaussian fit	VME digitizers	yes*	10 s/72 bunches
MR Tune monitor	Tune measurements	Ac Dia	Injection	1 kb	Fourier transform	GPIB signal analyzer	yes	5 sec
MR coupled Bunch	Coupled bunch mode meaurements	Ac Dia	User	10 kb	two-point correlation	GPIB digitizer	yes*	15 sec
MR coupling	Transverse coupling	Ac Dia	User	10 kb	cross correlation	GPIB signal analyzer	yes*	10 sec
MR kicker profile	Kicker profile	Ac Dia	Injection	1 kb	Waterfall plot	VXI digitizers	yes*	5 sec
Accumulator Beamloss	Pager Alarm when beam drops	Ac Dia	Cycle	1 kb	comparison	modem	yes	5 sec
Digital Damper	Damping of coupled bunch modes	Dev	Initialization	1 kb	None	VXI modules	no	inits hardware
MECAR	status display and filter download	Ac Dia	User	10 kb	filter calculation	NA	NA	1-5 sec
High voltage power supply	Testing HV supply	Mnt	User	10 kb	Plot	GPIB scope	yes	10 sec
Beam Position Modules	Testing of RF modules	Mnt	User	2 kb	signal response	GPIB sig gen & scope	yes	30 sec
Beam Position Detector	Testing of sensor	Mnt	User	2 kb	signal response	GPIB netwrk analyzer	yes	10 sec
Bunch Spacing	Bunch arrival times at detector	Mod	User	10 kb	addition of delays	NA	NA	< 1 sec
Sampled Bunch simulation	Model of system's signal proagation	Mod	User	10 kb	FFT	NA	NA	< 1 sec
* timer cards are made in-house	^ most time spent in GPIB transfers		# under development					

Table 2. Examples of Applications of the Instrumentation Platform

EXPERIENCE

Reliability

We have had excellent results with the reliability of the hardware platform and the software. Several nodes had uptimes of about half a year only to be shut down by a site-wide power outage. None of the desktop hardware has failed in the several years that nodes have been on-line. During the development phase, the programs could hang or crash, but once these bugs have been removed (often incompatibilities of extensions) the systems ran reliably. On a downside note, on every upgrade of the software (e.g., new versions LabVIEW or Mac OS) we found that problems arise that affect the reliability. Therefore we are very careful in upgrading our software, especially new versions of the development environment.

Code sharing and documentation

The ability of code sharing is often the pride of a new programming paradigm. How did we do with LabVIEW so far? Code for instruments that was made available by the companies was used quite a lot, not always right out of the box but often with small modifications to be suitable for our applications. The Acnet interface code was used in all on-line applications and did not require any modifications. Many of our applications share the same analysis routine, a non-linear gaussian fit. Initially, a fitting routine coming directly out of LabVIEW's analysis library was used but this was adapted to our use and improved in speed (see [3]). Other code used to store data on disk (circular buffers), or pop-up menus has been reused some but not yet as much as we would like. In these cases it was very hard to find common ground for the different applications.

Development

We did find that programming in LabVIEW enabled more people with different backgrounds to set up a system. The graphical nature of LabVIEW saves one from making the typical syntax errors in text-based language and thus saves time. One of the most powerful debugging features is that one can run each routine by itself interactively (without having to create a main program) or, when run as a part of a program, view the parameters and results, possibly in graphical format. One wonders why such a powerful debugging feature has not been built into text-based development environments where you always have to put a wrapper (main program) around the subroutine to test its operation.

Even though a LabVIEW program is a graphical diagram of data flow, it is still possible to do bad programming and make the code incomprehensible. Proper documentation and proper programming styles are still very important to complete the development with a maintainable system. We have put documentation on a WWW server to provide easy access and have been using it quite a lot ourselves during development. We are also standardizing the development cycle for LabVIEW systems. This is described in [8].

Software/hardware availability

As mentioned before, we were able to use a lot of the drivers supplied by the vendors of the instruments. In the case of the synchrotron light monitor, (see [3]), almost everything, including the software, was bought, from frame grabbers to high voltage power supplies, and was easily integrated using LabVIEW. In all cases of the on-line systems, the only in-house developed hardware were the timings cards (either CAMAC or GPIB) used with Fermilab-specific accelerator timing network. Because the platform uses a desktop computer, we were able to use utilities like screen snapshot takers, installer programs, and editors to help us document the development of the application.

The non-embedded computer

One of the advantages of having a non-embedded computer, but one that interfaces to the instrument or crate, is that it is easier to upgrade. The outdated desktop can still be used for many other purposes as long as speed doesn't matter. An embedded computer is rather a special purpose machine and it will be hard to find other uses. The desktop market also develops faster than the embedded computers and, because it a mass market, prices are typically lower. We were able to use desktops with PowerPC chips long before a PowerPC embedded computer was available. If your application uses VME or VXI modules, an embedded computer saves you space, but if you use a GPIB instrument and you don't need the crate other than to put the embedded computer in, a desktop computer will save you space.

One main issue to consider carefully is how one applies commercial products. If you buy a specific purpose computer, it is ikely that the company is geared to support your type of application. If you use a desktop computer and desktop type application, the more you use it out of the mainstream use, the less support you can expect. So not only do you get less support from the company (they make their money of their mainstream users) but also you will have to do more work yourself to implement the features you want to add to the mainstream use. We did not want to go further than to write an ACNET interface and ,at this point, have no plans to try and implement an application requiring a fast and deterministic feedback loop. The Macintosh OS and LabVIEW as well are not engineered for those purposes. We found

though, that this did not limit us too much, since much of the real-time operations can be done by the ever smarter instruments.

CONCLUSIONS

The LabVIEW/Macintosh combination with the addition of the ACNET interface has given us a general purpose instrumentation platform that is easy to use and able to implement a wide variety of applications, including those used for the continuous accelerator operation. We limited ourselves to applications that didn't require fast and deterministic feedback loops. Most of our applications provide results to operators, requiring only that the results are returned within the range of the operator's patience. We found that the integration of commercial software and hardware with ACNET enabled us to buy off-the-shelf products, thus reducing time and cost and making life a lot easier.

REFERENCES

[1] B. Peters, "Macintosh Parameter Page", (to be published).
[2] W. Blokland, "An Interface From LabVIEW To The Accelerator Controls Network", Proc. of Accelerator Inst. Workshop, Berkeley, USA, 1992, pp. 320-329.
[3] A. A. Hahn and P. Hurh, "Results From An Imaging Beam Monitor In The Tevatron Using Synchrotron Light", HEACC'92, Hamburg, Germany, July 1992, pp. 248-250.
[4] W. Blokland, "A VXI/LabVIEW-based Beamline Tuner", PAC'93, Washington, USA, 1993.
[5] E. Barsotti, "A longitudinal Bunch Monitoring System using LabVIEW and high-speed oscilloscopes", 1994 Accelerator Instrumentation Workshop, Vancouver, Canada, 1994, pp. 466-472.
[6] J. Zagel, "Booster Ion Profile Monitor using LabVIEW", 1994 Accelerator Instrumentation Workshop, Vancouver, Canada, 1994, pp. 384-390.
[7] J. Steimel, "Fast Digital Damper for the Fermilab Booster", PAC95, 1995.
[8] W. Blokland, "A LabVIEW-based Accelerator Instrumentation Platform", EPAC94, London, GB, 1994.